امنية المعلومات

5

Chapter 1

Information Security
(Principals of Cryptography)

Security

- Motivation: **Why do we need security?**
- Increased reliance on Information technology with or with out the use of networks.
- The use of IT has changed our lives drastically.
- We depend on E-mail, Internet banking, and several other governmental activities that use IT.
- Increased use of E-Commerce and the World wide web on the Internet as a vast repository of various kinds of information (immigration databases, flight tickets, stock markets etc.)

Security Concerns

- Systems connected by networks are more prone to attacks and also suffer more as a result of the attacks than stand-alone systems

- Concerns such as the following are common
  - How do I know the party I am talking on the network is really the one I want to talk to?
  - How can I be assured that no one else is listening and learning the data that I send over a network?
  - Can I ever stay relaxed that no hacker can enter my network and play havoc?

Concerns continued...

- Is the web site I am downloading information from a legitimate one, or a fake?

- How do I ensure that the person I just did a financial transaction denies having done it tomorrow or at a later time?

- I want to buy some thing online, but I don't want to let them charge my credit card before they deliver the product to me.

That is why...

- ..we need security
  - To safeguard the confidentiality, integrity, authenticity and availability of data transmitted over insecure networks
  - Internet is not the only insecure network in this world
  - Many internal networks in organizations are prone to insider attacks
  - In fact, insider attacks are greater both in terms of likelihood of happening and damage caused

What is information security?

confidentiality: only sender, intended receiver should "understand" message contents

- sender encrypts message
- receiver decrypts message

authentication: sender, receiver want to confirm identity of each other

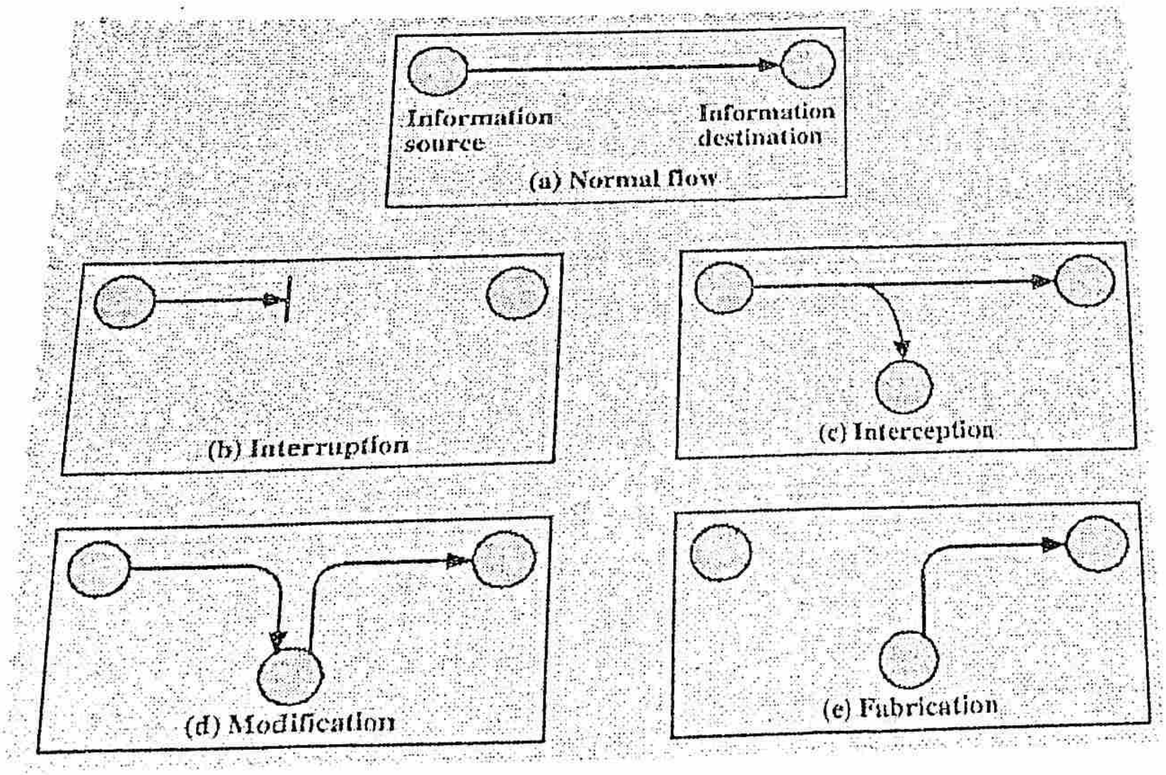
message integrity: sender, receiver want to ensure message not altered (in transit, or afterwards) without detection

access and availability: services must be accessible and available to users

Attacks, Services and Mechanisms

- **Security Attack:** Any action that threatens the security of information.
- **Security Mechanism:** A mechanism that is designed to detect, prevent, or recover from a security attack.
- **Security Service:** A service that enhances the security of data processing systems and information transfers. A security service makes use of one or more security mechanisms.

Security Attacks



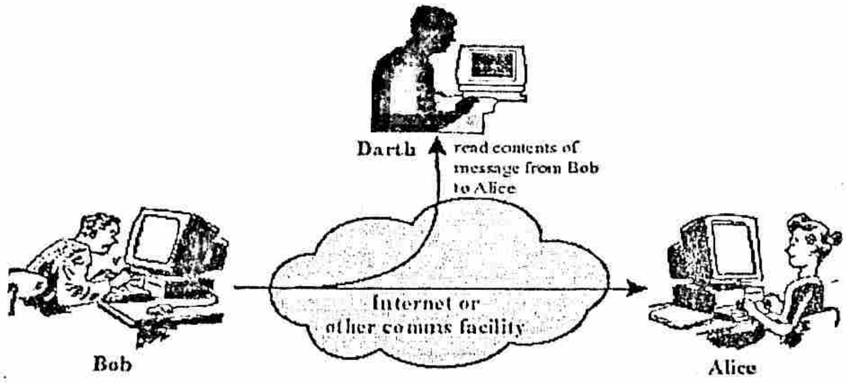
Security Attacks

- **Interruption:** This is an attack on availability
- **Interception:** This is an attack on confidentiality
- **Modfication:** This is an attack on integtrity
- **Fabrication:** This is an attack on authenticity

Passive and active attacks

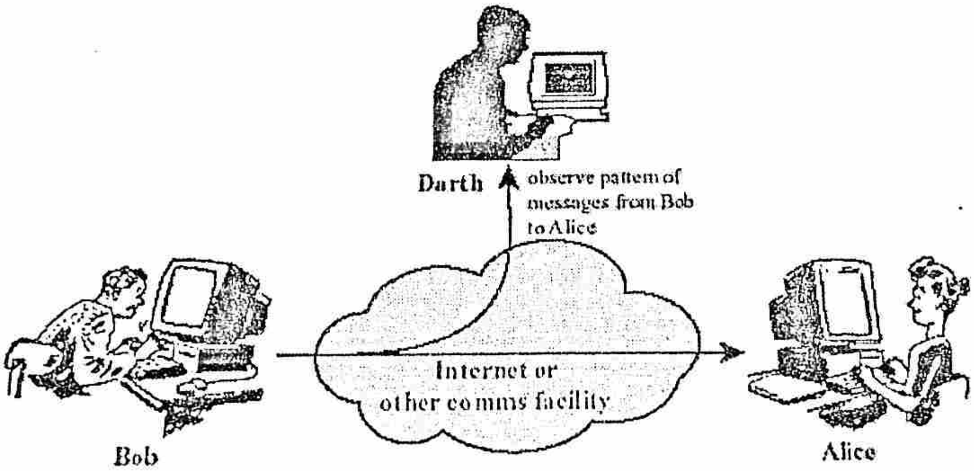
- **Passive attacks**
  - No modification of content or fabrication
  - Eavesdropping to learn contents or other information (transfer patterns, traffic flows etc.)
- **Active attacks**
  - Modification of content and/or participation in communication to
    - Impersonate legitimate parties
    - Modify the content in transit
    - Launch denial of service attacks

Passive Attacks



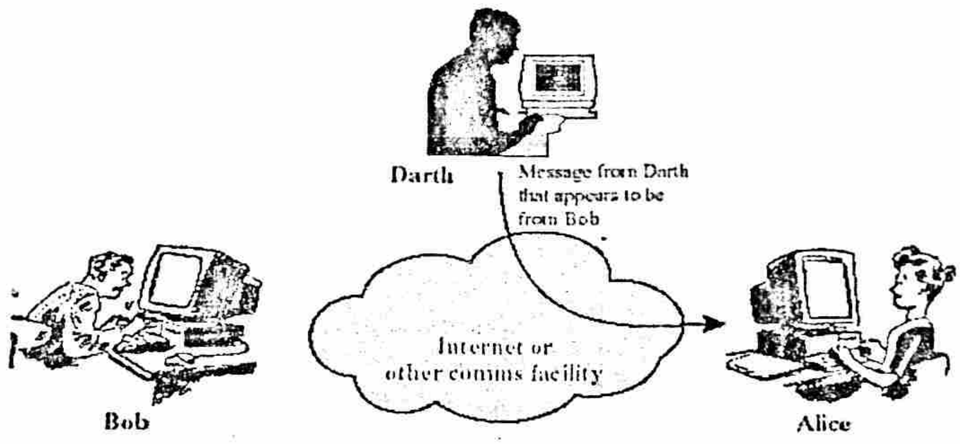
(a) Release of message contents

Passive Attacks



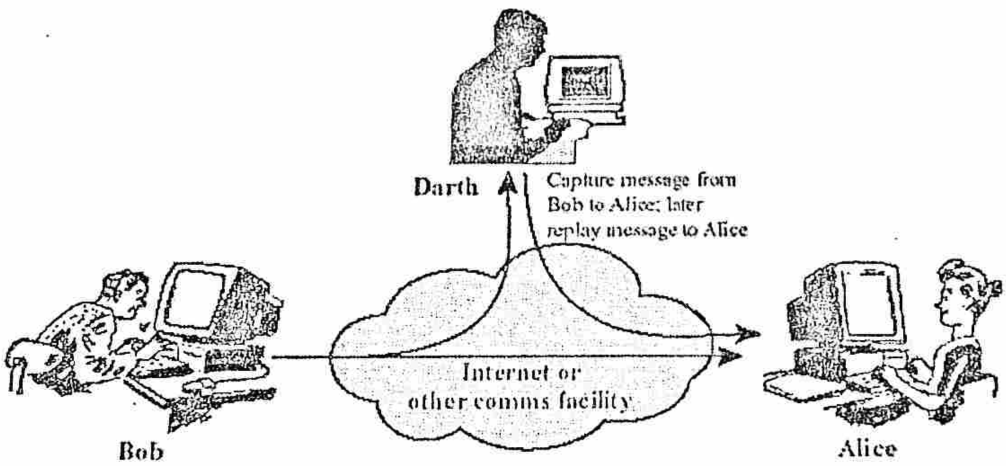
(b) Traffic analysis

Active Attacks



(a) Masquerade

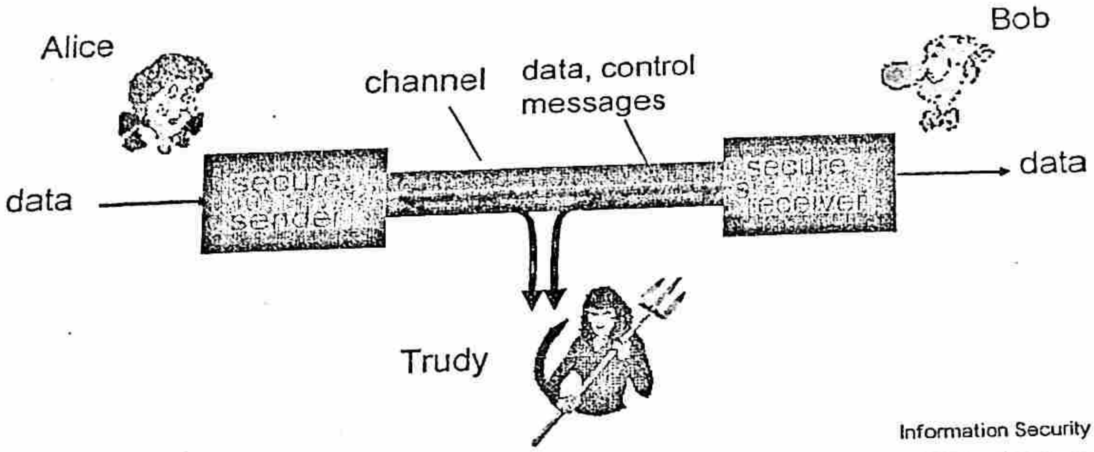
Active Attacks



(b) Replay

Friends and enemies: Alice, Bob, Trudy

- well-known in network security world
- Bob, Alice want to communicate "securely"
- Trudy (intruder) may intercept, delete, add messages



Who might Bob, Alice be?

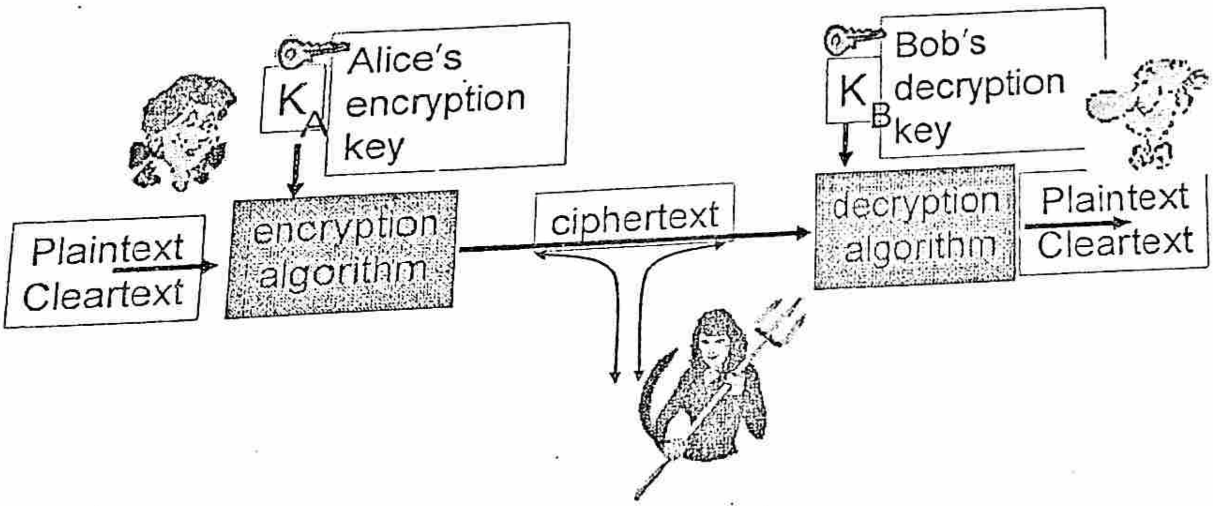
- ... well, *real-life* Bobs and Alices!
- Web browser/server for electronic transactions (e.g., on-line purchases)
- on-line banking client/server
- DNS servers
- routers exchanging routing table updates

There are bad guys (and girls) out there!

Q: What can a "bad guy" do?

- *eavesdrop:* intercept messages
- actively *insert* messages into connection
- *impersonation:* can fake (spoof) source address in packet (or any field in packet)
- *hijacking:* "take over" ongoing connection by removing sender or receiver, inserting himself in place
- *denial of service*: prevent service from being used by others (e.g., by overloading resources)

The language of cryptography



m plaintext message

$K_A(m)$ ciphertext, encrypted with key K_A

$m = K_B(K_A(m))$

Simple encryption scheme

substitution cipher: substituting one thing for another

- Caesar Cipher: taking each letter in the plain text message and substituting the letter that is k letters later.
- monoalphabetic cipher: substitute one letter for another

```
plaintext:   abcdefghijklmnopqrstuvwxyz

ciphertext:  mnbvcxzasdfghjklpoiuytrewq
```

e.g.:
```
Plaintext: bob. alice
ciphertext: nkn. mgsbc
```

Encryption key: mapping from set of 26 letters to set of 26 letters

Breaking an encryption scheme

- cipher-text only attack: Trudy has ciphertext she can analyze
- two approaches:
  - brute force: search through all keys
  - statistical analysis
- known-plaintext attack: Trudy has plaintext corresponding to ciphertext
  - e.g., in monoalphabetic cipher, Trudy determines pairings for a,l,i,c,e,b,o,
- chosen-plaintext attack: Trudy can get ciphertext for chosen plaintext

Simple encryption scheme

Polyalphabetic Encryption: using multiple momoalphabetic ciphers

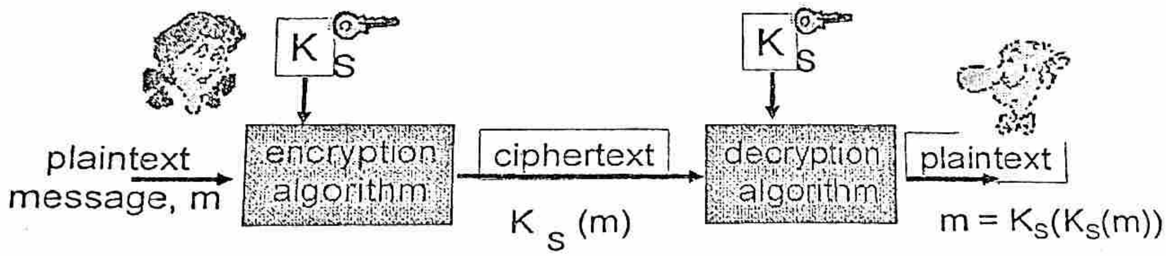
- The same letter appearing in different positions in the plaintext might be encoded differently.
- We might choose to use these two Caesar ciphers C1 and C2 in the repeating pattern C1, C2, C2, C1, C2.

Plaintext letter:	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
$C_1(k = 5)$:	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e
$C_2(k = 19)$:	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s

e.g.:

```
Plaintext: bob. alice
ciphertext: ghu. fenvx
```


Symmetric key cryptography



symmetric key crypto: Bob and Alice share same (symmetric) key: K_S

- e.g., key is knowing substitution pattern in mono alphabetic substitution cipher

Block Ciphers

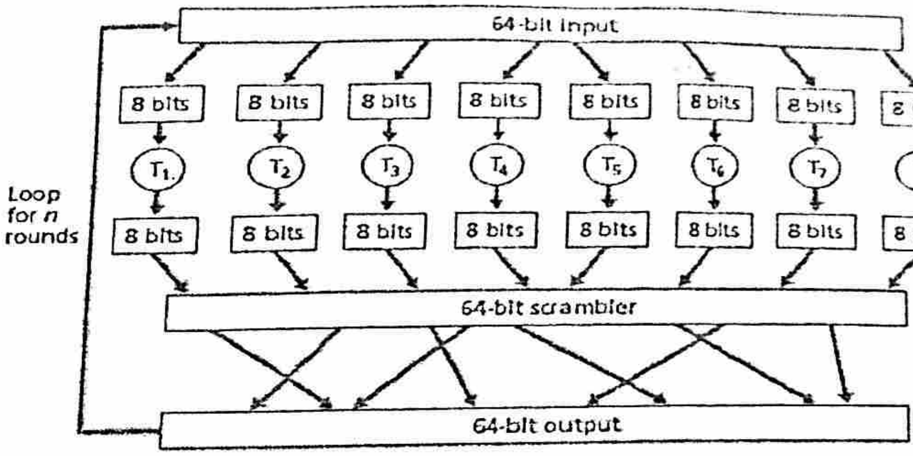
- There are two broad classes of symmetric encryption techniques: **stream ciphers** and **block ciphers.**
- Stream ciphers is used for wireless LAN.
- block ciphers are used in many secure Internet protocols, including PGP (for secure e-mail), SSL (for securing TCP connections), and IPsec (for securing the network-layer transport).

Block Ciphers

- The message to be encrypted is processed in blocks of k bits.
- Each block is encrypted independently.
- An example, suppose that k=3, one possible mapping is given in below table.
- The message 010110001111 gets encrypted into 101000111001.

input	output	input	output
000	110	100	011
001	111	101	010
010	101	110	000
011	100	111	001

An example of a block cipher



Symmetric key crypto: DES

DES: Data Encryption Standard

- US encryption standard [NIST (National Institute of Standards and Technology) 1993]
- 56-bit symmetric key (An algorithm's key determines the specific "mini-table" mappings), 64-bit plaintext input
- making DES more secure: 3DES

block cipher with cipher block chaining

- Important problem occurs when two or more of the clear text blocks can be identical.
- Example: the clear text in two or more blocks could be "HTTP/1.1". For these identical blocks, a block cipher would, of course, produce the same cipher text. An attacker could potentially guess the clear text when it sees identical cipher text blocks.
- Solution: mix some randomness into the cipher text so that identical plaintext blocks produce different cipher text blocks.

- Example: consider a 3-bit block cipher. Suppose the plaintext is 010010010. (0XOR1= 1XOR1=0)
- Direct encryption 101101101. Easy to know that the three plaintexts are the same.
- Random blocks r(1) = 001, r(2) =111, and r(3) = 100.
- Cipher text (c(i) = KS(m(i) XOR r(i))) c(1) = 011, c(2) = 101, and c(3) = 110.
- Note that the three Cipher text blocks are different even though the plaintext blocks are the same.
- We must transmit twice as many bits as before, doubling the required bandwidth.

Cipher Block Chaining (CBC)

- 1. Before encrypting the message, the sender generates a random k-bit string, called the Initialization Vector (IV). Denote this initialization vector by c(0). The sender sends the IV to the receiver in clear text.
- 2. For the first block, the sender calculate c(1) = KS(m(1) XOR c(0)).
- 3. For the ith block, the sender generates the ith cipher text block from c(i) = KS(m(i) XOR c(i - 1)).

- Example: 3-bit block cipher, plaintext 010010010 and IV = c(0) = 001.
- c(1) = KS(m(1) XOR c(0)) = 011.
- c(2) = KS(m(2) XOR c(1)) = KS(010 XOR 011) = 001,
- c(3) = KS(m(3) XOR c(2)) = KS(010 XOR 001) = 011.

AES: Advanced Encryption Standard

- symmetric-key NIST standard, replacied DES (Nov 2001)
- processes data in 128 bit blocks
- 128, 192, or 256 bit keys
- brute force decryption (try each key) taking 1 sec on DES, takes 149 trillion years for AES

Public Key Cryptography

symmetric key crypto

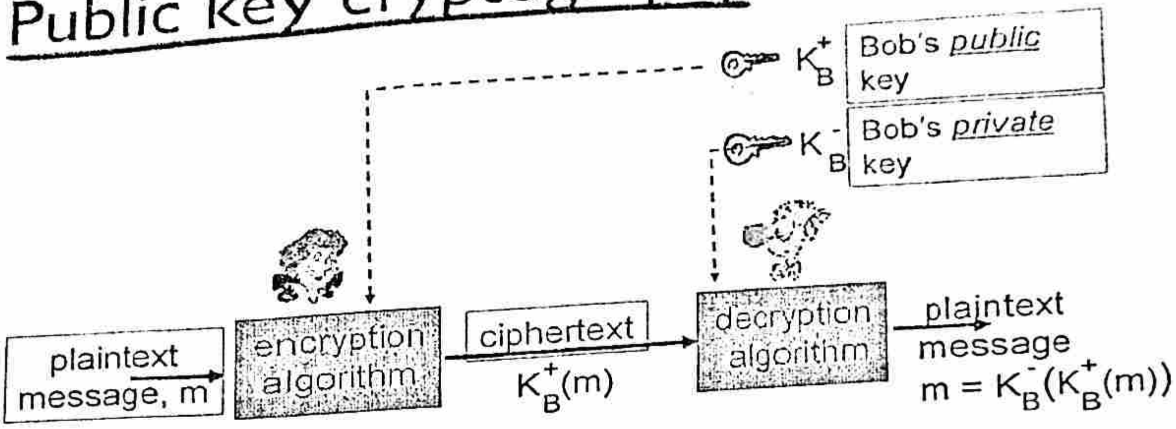
- requires sender, receiver know shared secret key
- Q: how to agree on key in first place (particularly if never "met")?

public key crypto

- radically different approach [Diffie-Hellman76, RSA78]
- sender, receiver do *not* share secret key
- *public* encryption key known to *all*
- *private* decryption key known only to receiver



Public key cryptography



two immediate worries

- the intruder knows both the public key and the algorithm that is used for encryption
- Since encryption key is public, anyone can send an encrypted message to Bob

Public key encryption algorithms

requirements:

(1) need $K_B^+(\cdot)$ and $K_B^-(\cdot)$ such that

$$K_B^-(K_B^+(m)) = m$$

(2) given public key K_B^+, it should be impossible to compute private key K_B^-

RSA: Rivest, Shamir, Adelson algorithm

Prerequisite: modular arithmetic

- x mod n = remainder of x when divide by n
- facts:
  [(a mod n) + (b mod n)] mod n = (a+b) mod n
  [(a mod n) - (b mod n)] mod n = (a-b) mod n
  [(a mod n) * (b mod n)] mod n = (a*b) mod n
- thus
  $(a \bmod n)^d \bmod n = a^d \bmod n$
- example: x=14, n=10, d=2:
  $(x \bmod n)^d \bmod n = 4^2 \bmod 10 = 6$
  $x^d = 14^2 = 196 \quad x^d \bmod 10 = 6$

RSA: getting ready

- message: just a bit pattern
- bit pattern can be uniquely represented by an integer number
- thus, encrypting a message is equivalent to encrypting a number.

example:

- m= 10010001 . This message is uniquely represented by the decimal number 145.
- to encrypt m, we encrypt the corresponding number, which gives a new number (the ciphertext).

RSA: Creating public/private key pair

1. choose two large prime numbers p, q. (e.g., 1024 bits each)
2. compute $n = pq$, $z = (p-1)(q-1)$
3. choose e (with $e<n$) that has no common factors with z (e, z are "relatively prime").
4. choose d such that $ed-1$ is exactly divisible by z. (in other words: $ed \bmod z = 1$).
5. *public* key is $\underbrace{(n,e)}_{K_B^+}$. *private* key is $\underbrace{(n,d)}_{K_B^-}$.

RSA: encryption, decryption

0. given (n,e) and (n,d) as computed above
1. to encrypt message m ($<n$), compute
   $c = m^e \bmod n$
2. to decrypt received bit pattern, c, compute
   $m = c^d \bmod n$

magic happens! $m = \underbrace{(m^e \bmod n)}_{c}{}^d \bmod n$

RSA example:

Bob chooses $p=5$, $q=7$. Then $n=35$, $z=24$.
$e=5$ (so e, z relatively prime).
$d=29$ (so $ed-1$ exactly divisible by z).

encrypting 8-bit messages.

	Original m	m	m^e	$c = m^e \bmod n$
encrypt:	L	12	24832	17

	c	c^d	$m = c^d \bmod n$
decrypt:	17	481968572106750915091411825223071697	12

Why does RSA work?

- must show that $c^d \bmod n = m$
  where $c = m^e \bmod n$
- fact: for any x and y: $x^y \bmod n = x^{(y \bmod z)} \bmod n$
  - where $n = pq$ and $z = (p-1)(q-1)$
- thus,

$$c^d \bmod n = (m^e \bmod n)^d \bmod n$$
$$= m^{ed} \bmod n$$
$$= m^{(ed \bmod z)} \bmod n$$
$$= m^1 \bmod n$$
$$= m$$

RSA: another important property

The following property will be *very* useful later:

$$K_B^-(K_B^+(m)) = m = K_B^+(K_B^-(m))$$

use public key first, followed by private key

use private key first, followed by public key

result is the same!

Why $K_B^-(K_B^+(m)) = m = K_B^+(K_B^-(m))$?

follows directly from modular arithmetic:

$$\begin{aligned}(m^e \bmod n)^d \bmod n &= m^{ed} \bmod n \\ &= m^{de} \bmod n \\ &= (m^d \bmod n)^e \bmod n\end{aligned}$$

Why is RSA secure?

- suppose you know Bob's public key (n,e). How hard is it to determine d?
- essentially need to find factors of n without knowing the two factors p and q
  - fact: factoring a big number is hard

RSA in practice: session keys

- exponentiation in RSA is computationally intensive
- DES is at least 100 times faster than RSA
- use public key cryto to establish secure connection, then establish second key – symmetric session key – for encrypting data

session key, K_S

- Bob and Alice use RSA to exchange a symmetric key K_S
- once both have K_S, they use symmetric key cryptography

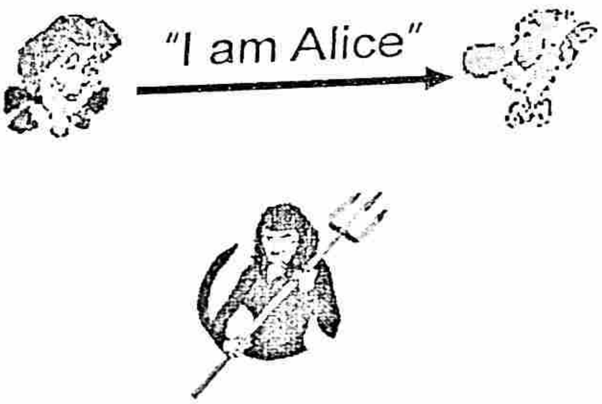
Chapter 2

Network Security (Message integrity, *authentication)*

Authentication

Goal: Bob wants Alice to "prove" her identity to him

Protocol ap1.0: Alice says "I am Alice"



Failure scenario??

Authentication

Goal: Bob wants Alice to "prove" her identity to him

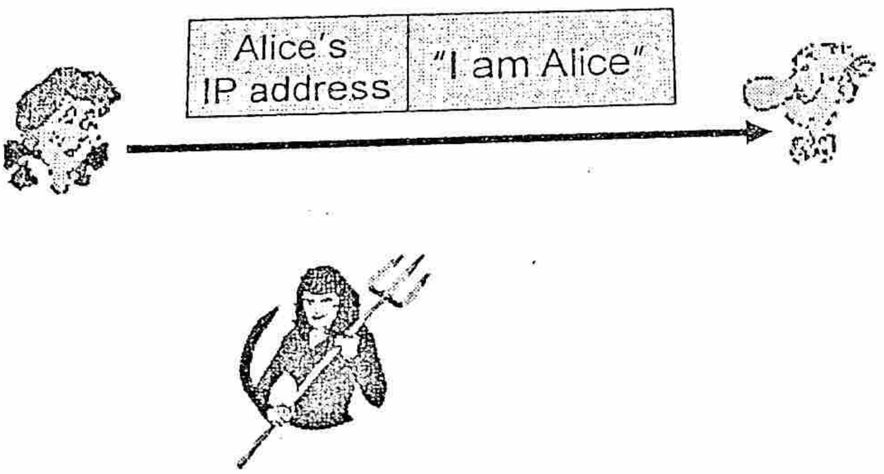
Protocol ap1.0: Alice says "I am Alice"



in a network,
Bob can not "see" Alice,
so Trudy simply declares
herself to be Alice

Authentication: another try

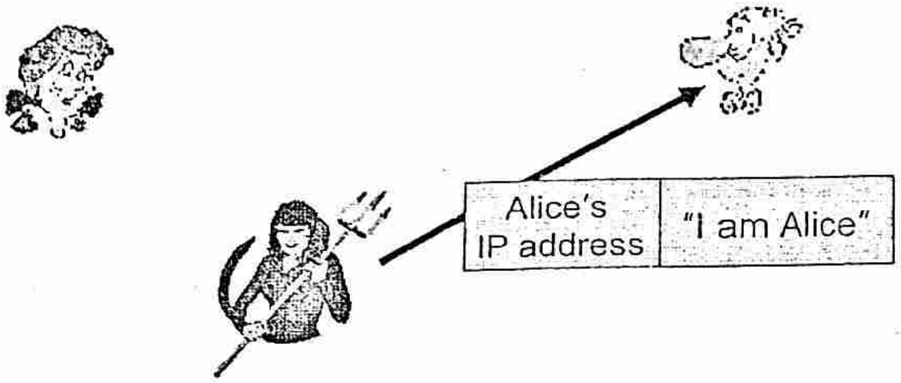
Protocol ap2.0: Alice says "I am Alice" in an IP packet containing her source IP address



Failure scenario??

Authentication: another try

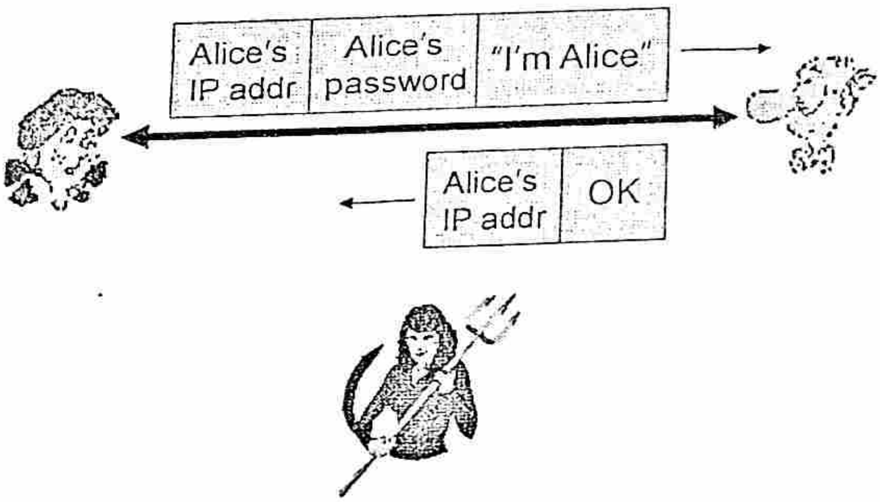
Protocol ap2.0: Alice says "I am Alice" in an IP packet containing her source IP address



Trudy can create a packet "spoofing" Alice's address

Authentication: another try

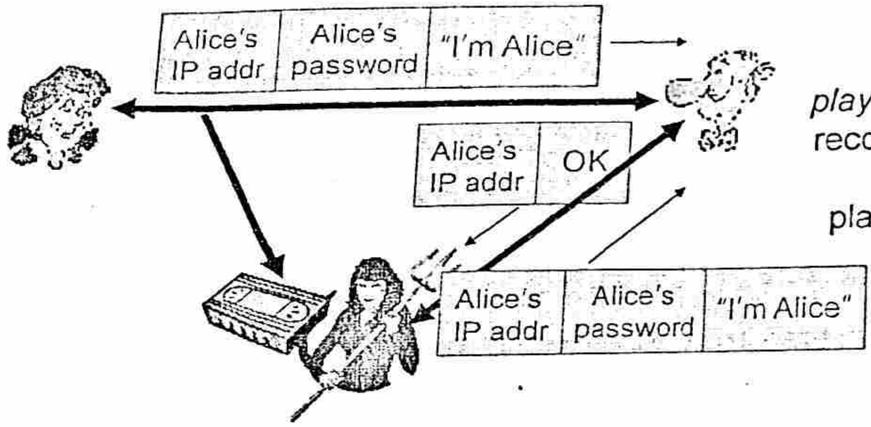
Protocol ap3.0: Alice says "I am Alice" and sends her secret password to "prove" it.



Failure scenario??

Authentication: another try

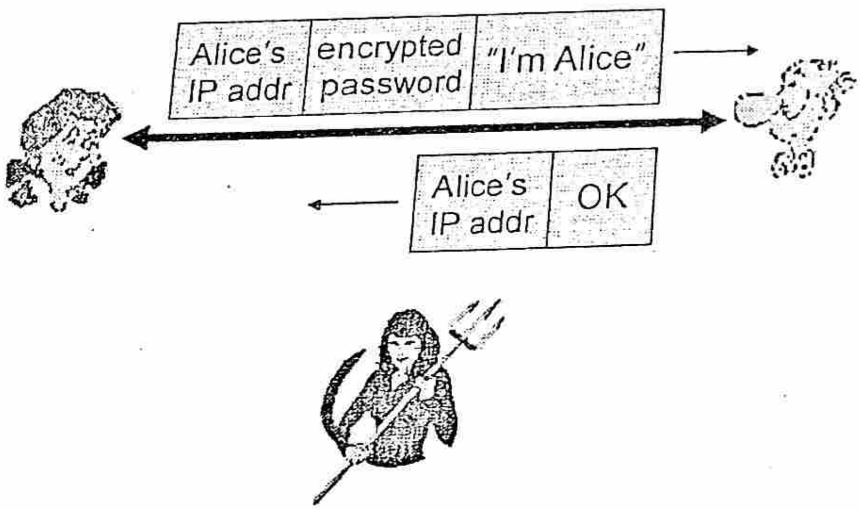
Protocol ap3.0: Alice says "I am Alice" and sends her secret password to "prove" it.



playback attack: Trudy records Alice's packet and later plays it back to Bob

Authentication: yet another try

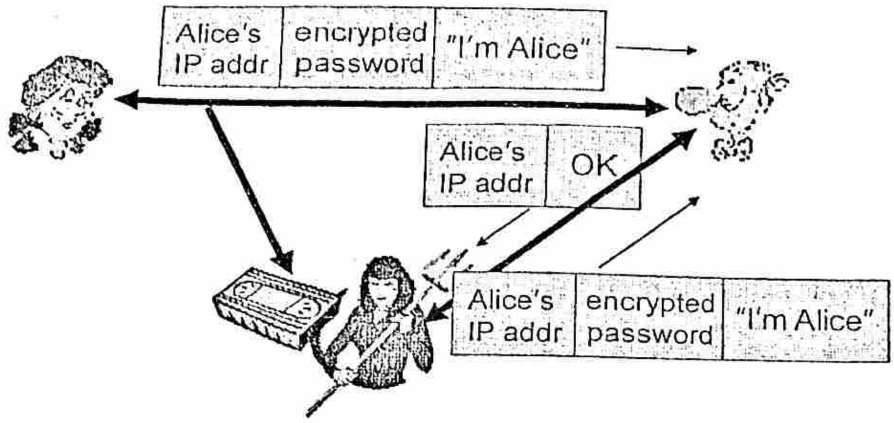
Protocol ap3.1: Alice says "I am Alice" and sends her *encrypted* secret password to "prove" it.



Failure scenario??

Authentication: yet another try

Protocol ap3.1: Alice says "I am Alice" and sends her *encrypted* secret password to "prove" it.



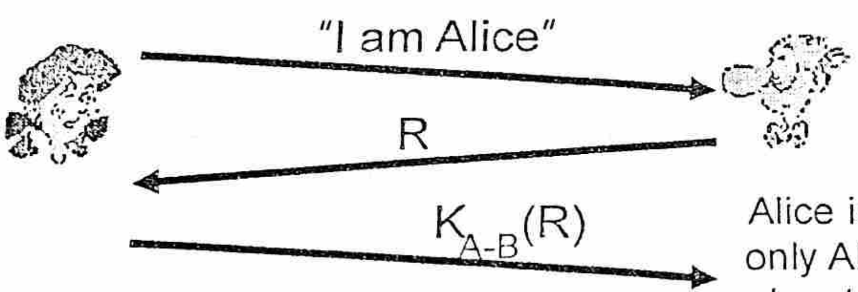
record and playback *still* works!

Authentication: yet another try

Goal: avoid playback attack

nonce: number (R) used only *once-in-a-lifetime*

ap4.0: to prove Alice "live", Bob sends Alice *nonce*, R. Alice must return R, encrypted with shared secret key



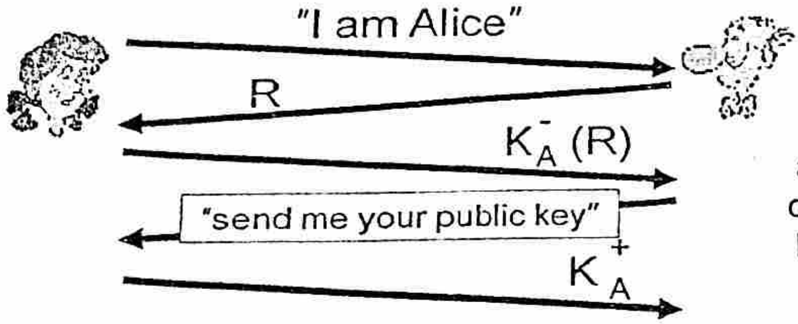
Alice is live, and only Alice knows key to encrypt nonce, so it must be Alice!

Failures, drawbacks?

Authentication: ap5.0

ap4.0 requires shared symmetric key
- can we authenticate using public key techniques?

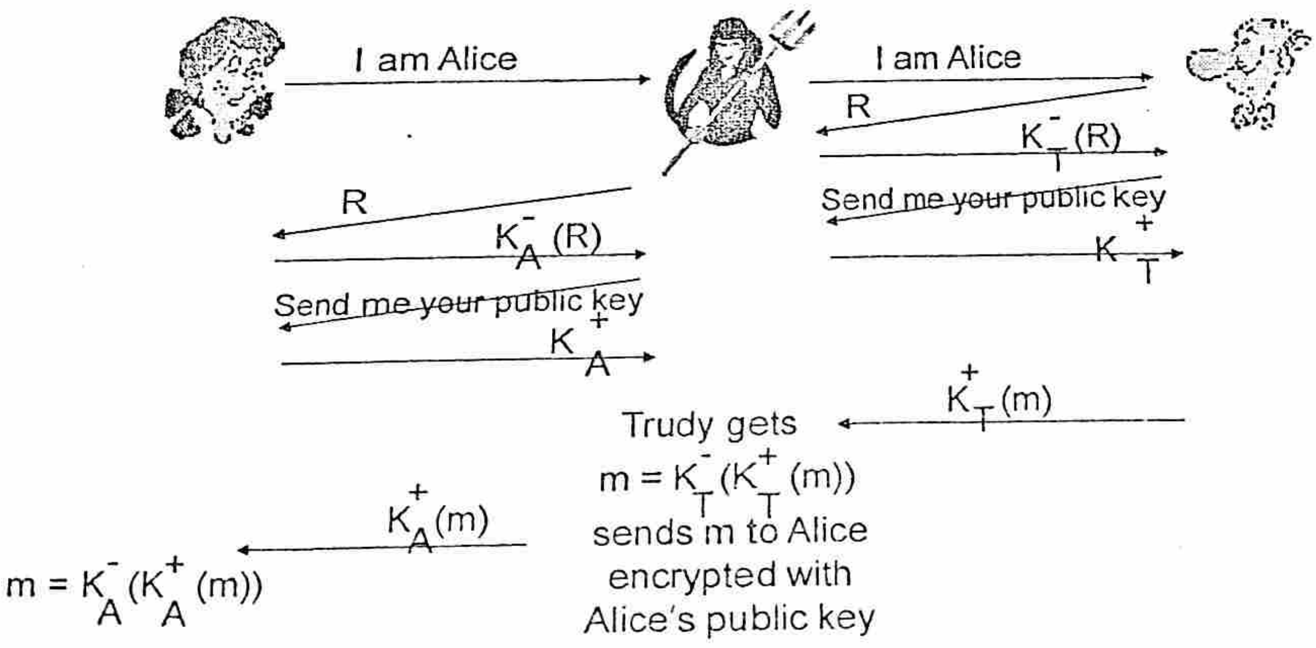
ap5.0: use nonce, public key cryptography



Bob computes
$K_A^+(K_A^-(R)) = R$
and knows only Alice could have the private key, that encrypted R such that
$K_A^+(K_A^-(R)) = R$

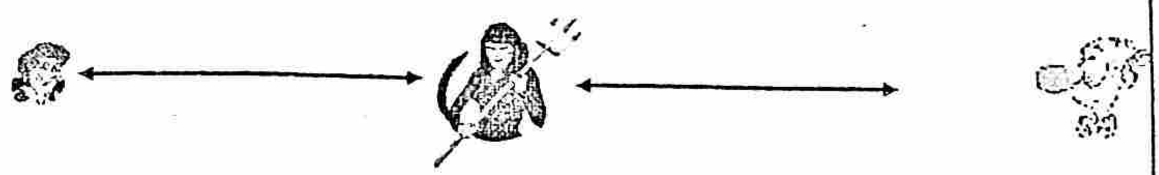
ap5.0: security hole

man (or woman) in the middle attack: Trudy poses as Alice (to Bob) and as Bob (to Alice)



ap5.0: security hole

man (or woman) in the middle attack: Trudy poses as Alice (to Bob) and as Bob (to Alice)



difficult to detect:

- Bob receives everything that Alice sends, and vice versa. (e.g., so Bob, Alice can meet one week later and recall conversation!)
- problem is that Trudy receives all messages as well!

Digital signatures

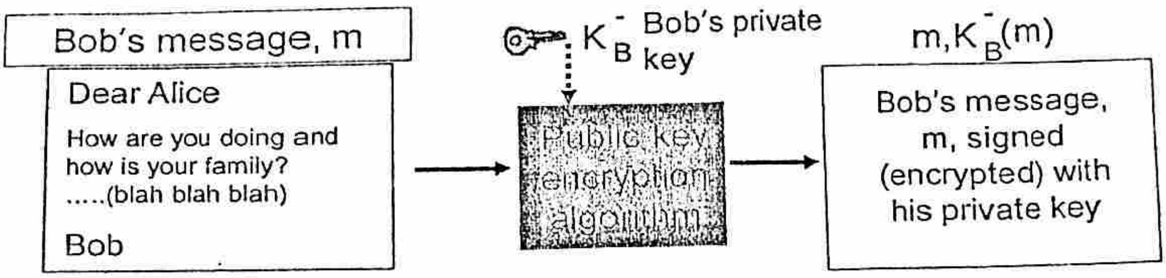
cryptographic technique analogous to hand-written signatures:

- sender (Bob) digitally signs document, establishing he is document owner/creator.
- *verifiable, nonforgeable:* recipient (Alice) can prove to someone that Bob, and no one else (including Alice), must have signed document

Digital signatures

simple digital signature for message m:

- Bob signs m by encrypting with his private key K_B^-, creating "signed" message, $K_B^-(m)$



Digital signatures

- suppose Alice receives msg m, with signature: $m, K_B^-(m)$
- Alice verifies m signed by Bob by applying Bob's public key K_B^+ to $K_B^-(m)$ then checks $K_B^+(K_B^-(m)) = m$.
- If $K_B^+(K_B^-(m)) = m$, whoever signed m must have used Bob's private key.

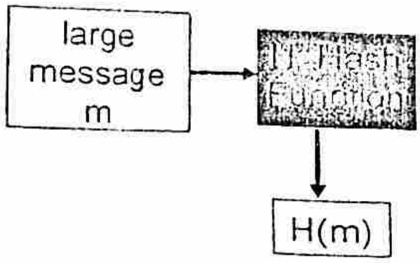
Alice thus verifies that:

- ✓ Bob signed m
- ✓ no one else signed m
- ✓ Bob signed m and not m' (altered data)

non-repudiation:

- ✓ Alice can take m, and signature $K_B^-(m)$ to court and prove that Bob signed m

Message digests



computationally expensive to public-key-encrypt long messages

goal: fixed-length, easy- to-compute digital "fingerprint"

- apply hash function H to *m*, get fixed size message digest, *H(m)*.

Hash function properties:

- many-to-1
- produces fixed-size msg digest (fingerprint)
- given message digest x, computationally infeasible to find m such that x = H(m)

Internet checksum: poor crypto hash function

Internet checksum has some properties of hash function:

✓ produces fixed length digest (16-bit sum) of message

✓ is many-to-one

But given message with given hash value, it is easy to find another message with same hash value:

message	ASCII format
I O U 1	49 4F 55 31
0 0 . 9	30 30 2E 39
9 B O B	39 42 D2 42
	B2 C1 D2 AC

message	ASCII format
I O U 9	49 4F 55 39
0 0 . 1	30 30 2E 31
9 B O B	39 42 D2 42
	B2 C1 D2 AC

different messages but identical checksums!

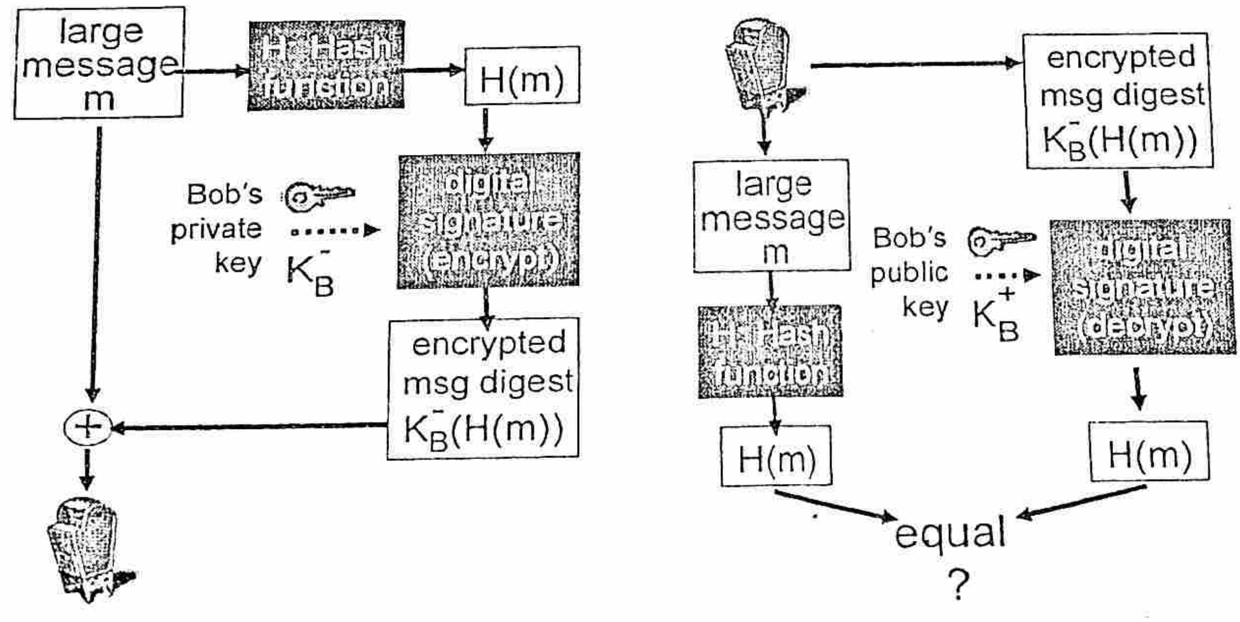
Hash function algorithms

- MD5 hash function widely used
  - computes 128-bit message digest in 4-step process.
  - arbitrary 128-bit string x, appears difficult to construct msg m whose MD5 hash is equal to x
- SHA-1 is also used
  - US standard [NIST, FIPS PUB 180-1]
  - 160-bit message digest

Digital signature = signed message digest

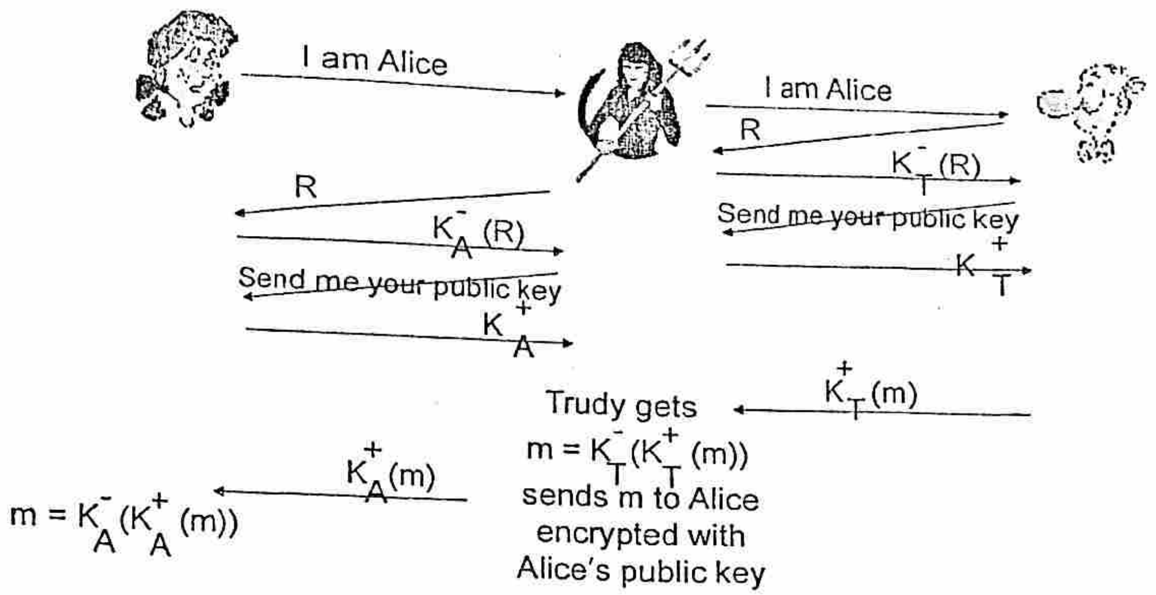
Bob sends digitally signed message:

Alice verifies signature, integrity of digitally signed message:



Recall: ap5.0 security hole

man (or woman) in the middle attack: Trudy poses as Alice (to Bob) and as Bob (to Alice)

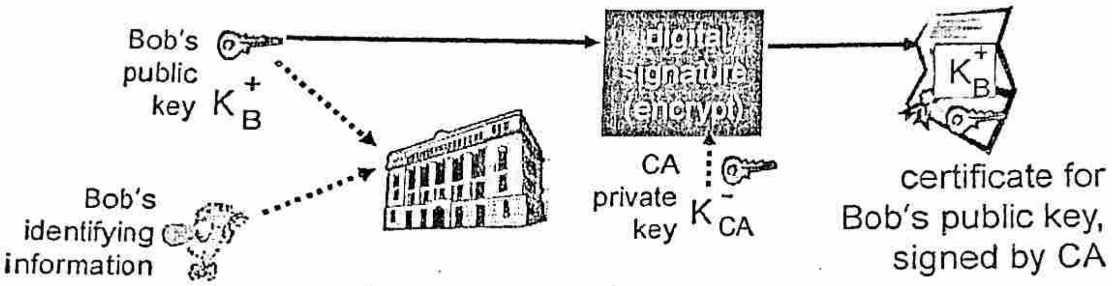


Public-key certification

- motivation: Trudy plays pizza prank on Bob
  - Trudy creates e-mail order:
    Dear Pizza Store, Please deliver to me four pepperoni pizzas. Thank you, Bob
  - Trudy signs order with her private key
  - Trudy sends order to Pizza Store
  - Trudy sends to Pizza Store her public key, but says it's Bob's public key
  - Pizza Store verifies signature; then delivers four pepperoni pizzas to Bob
  - Bob doesn't even like pepperoni

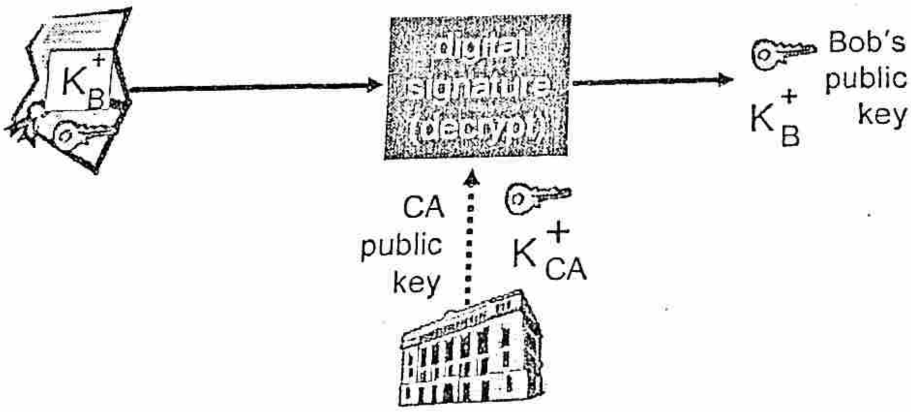
Certification authorities

- *certification authority (CA):* binds public key to particular entity, E.
- E (person, router) registers its public key with CA.
  - E provides "proof of identity" to CA.
  - CA creates certificate binding E to its public key.
  - certificate containing E's public key digitally signed by CA – CA says "this is E's public key"



Certification authorities

- when Alice wants Bob's public key:
  - gets Bob's certificate (Bob or elsewhere).
  - apply CA's public key to Bob's certificate, get Bob's public key

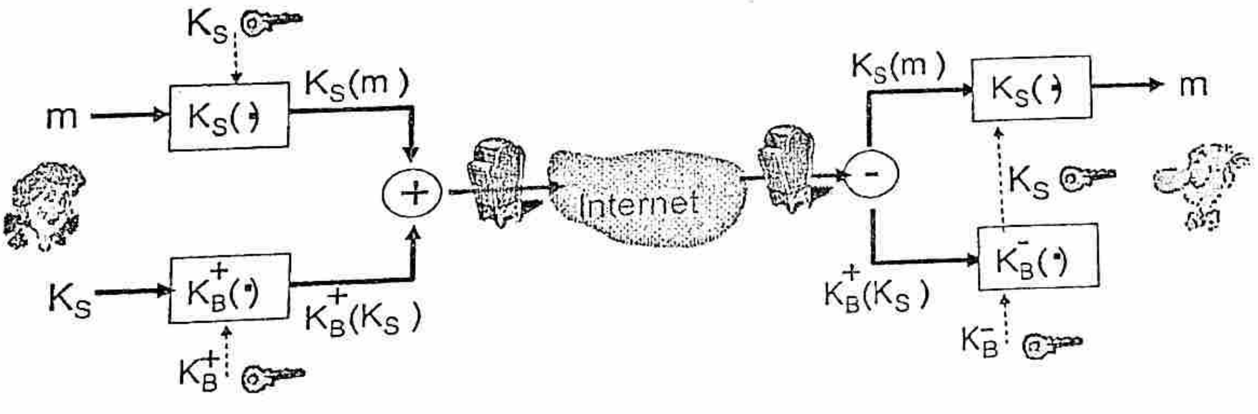


Chapter 3

Network Security (Securing e-mail and TCP connections)

Secure e-mail

- Alice wants to send confidential e-mail, m, to Bob.

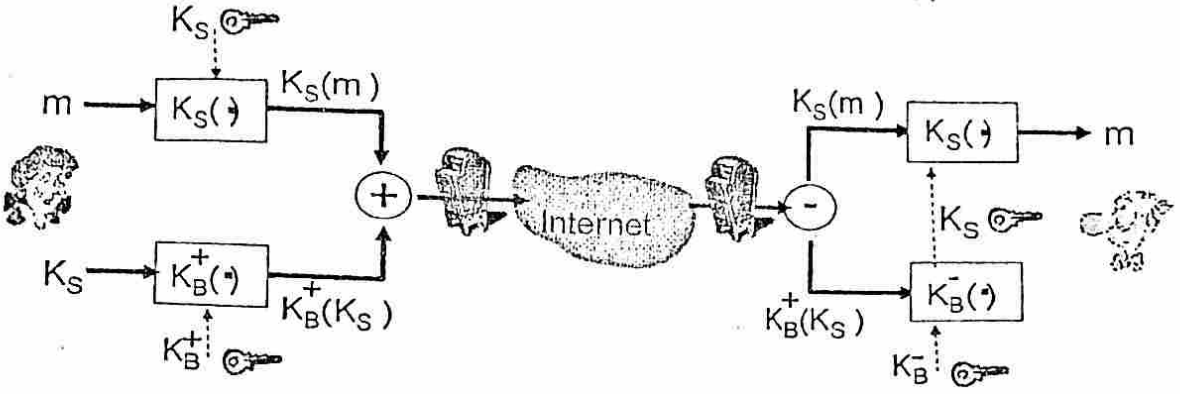


Alice:

- generates random *symmetric* private key, K_S
- encrypts message with K_S (for efficiency)
- also encrypts K_S with Bob's public key
- sends both $K_S(m)$ and $K_B(K_S)$ to Bob

Secure e-mail

- Alice wants to send confidential e-mail, m, to Bob.

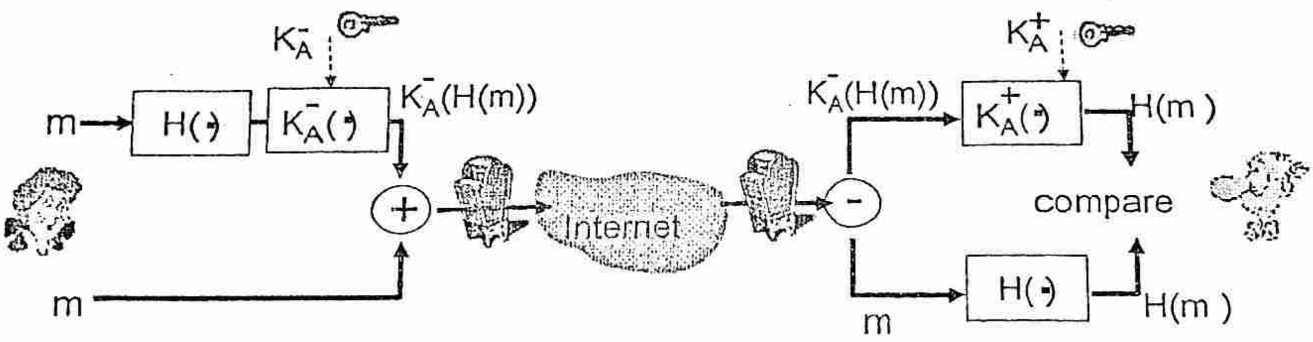


Bob:

- uses his private key to decrypt and recover K_S
- uses K_S to decrypt $K_S(m)$ to recover m

Secure e-mail (continued)

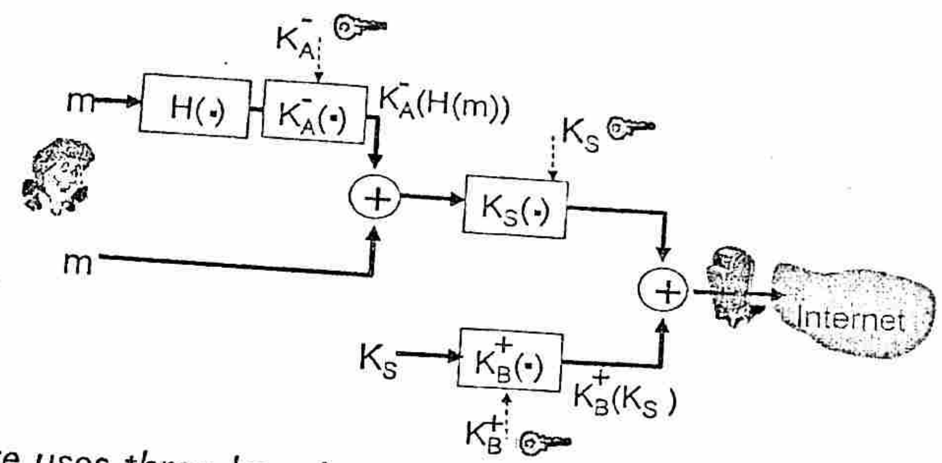
- Alice wants to provide sender authentication message integrity



- Alice digitally signs message
- sends both message (in the clear) and digital signature

Secure e-mail (continued)

❖ Alice wants to provide secrecy, sender authentication, message integrity.



❖ *Alice uses three keys:* her private key, Bob's public key, newly created symmetric key

❖ Pretty Good Privacy (PGP) is an e-mail encryption scheme that has become a de facto standard.

(SSL: Secure Sockets Layer)

❖ widely deployed security protocol
  - supported by almost all browsers, web servers
  - *https*
  - *billions $/year over SSL*

❖ mechanisms: [Woo 1994], implementation: Netscape

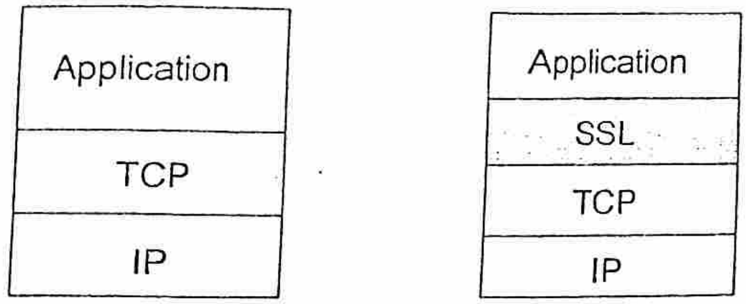
❖ variation -TLS: *transport layer security, RFC 2246*

❖ provides
  - *confidentiality*
  - *integrity*
  - *authentication*

❖ original goals:
  - Web e-commerce transactions
  - encryption (especially credit-card numbers)
  - Web-server authentication
  - optional client authentication
  - minimum hassle in doing business with new merchant

❖ available to all TCP applications
  - secure socket interface

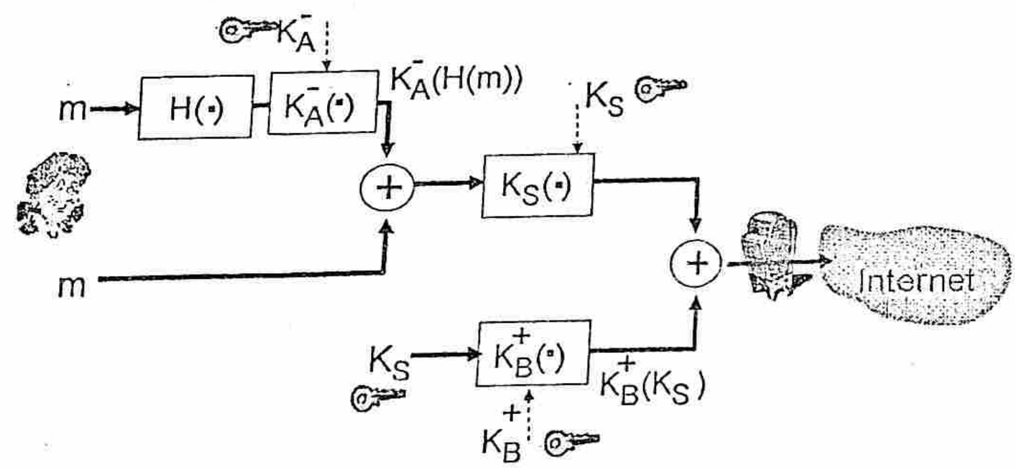
SSL and TCP/IP



normal application *application with SSL*

- SSL provides application programming interface (API) to applications
- C and Java SSL libraries/classes readily available

Could do something like PGP:

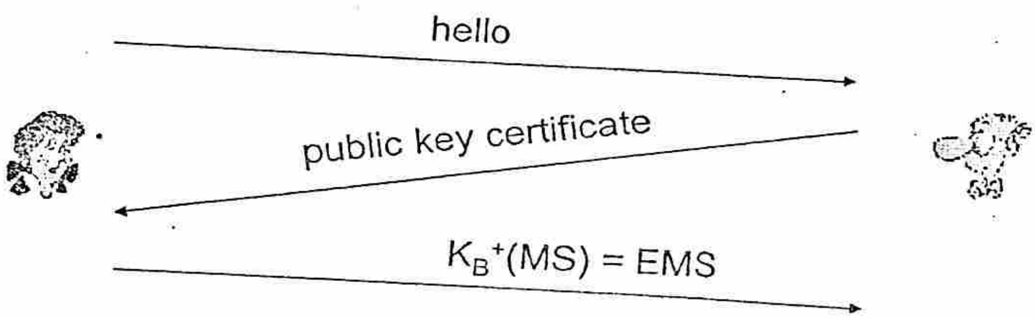


- but want to send byte streams & interactive data
- want set of secret keys for entire connection
- want certificate exchange as part of protocol: handshake phase

Almost SSL: a simple secure channel

- *handshake:* Alice and Bob use their certificates, private keys to authenticate each other and exchange shared secret
- *key derivation:* Alice and Bob use shared secret to derive set of keys
- *data transfer:* data to be transferred is broken up into series of records
- *connection closure:* special messages to securely close connection

A simple handshake



MS: master secret

EMS: encrypted master secret

key derivation

- considered bad to use same key for more than one cryptographic operation
  - use different keys for message authentication code (MAC) and encryption
- four keys:
  - K_c = encryption key for data sent from client to server
  - M_c = MAC key for data sent from client to server
  - K_s = encryption key for data sent from server to client
  - M_s = MAC key for data sent from server to client
- keys derived from key derivation function (KDF)
  - takes master secret and (possibly) some additional random data and creates the keys

Data records

- why not encrypt data in constant stream as we write it to TCP?
  - where would we put the MAC? If at end, no message integrity until all data processed.
  - e.g., with instant messaging, how can we do integrity check over all bytes sent before displaying?
- instead, break stream in series of records
  - each record carries a MAC = H(Data record + M_c)
  - Encrypted Data = K_c (Data + MAC)
  - receiver can act on each record as it arrives



Sequence numbers

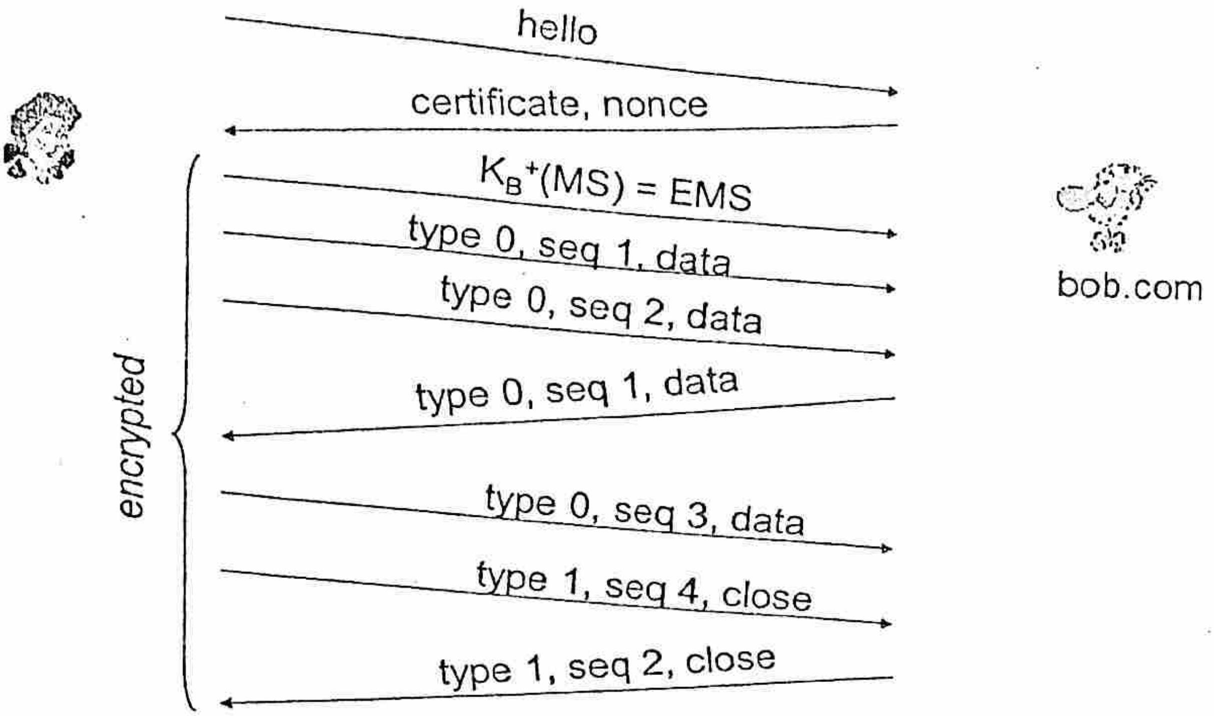
- *problem:* attacker can capture and replay record or re-order records
- *solution:* put sequence number into MAC:
  - MAC = H(Record data + M_c + Sequence number)
  - note: no sequence number field
- *problem:* attacker could replay all records

Control Information

- *problem:* truncation attack:
  - attacker forges TCP connection close segment
  - one or both sides thinks there is less data than there actually is.
- *solution:* record types, with one type for closure
  - type 0 for data; type 1 for closure
- MAC = MAC(Record data + M_c + Sequence number + type)

length	type	data	MAC

SSL: summary



Almost SSL isn't complete

- how long are fields?
- which encryption protocols?
- want negotiation?
  - allow client and server to support different encryption algorithms
  - allow client and server to choose together specific algorithm before data transfer

SSL cipher suite

- cipher suite
  - public-key algorithm
  - symmetric encryption algorithm
  - MAC algorithm
- SSL supports several cipher suites
- negotiation: client, server agree on cipher suite
  - client offers choice
  - server picks one

common SSL symmetric ciphers
- DES – Data Encryption Standard: block
- 3DES – Triple strength: block
- RC2 – Rivest Cipher 2: block
- RC4 – Rivest Cipher 4: stream

SSL Public key encryption
- RSA

Real SSL: handshake (I)

Purpose

1. server authentication
2. negotiation: agree on crypto algorithms
3. establish keys
4. client authentication (optional)

Real SSL: handshake (2)

1. client sends list of algorithms it supports, along with client nonce
2. server chooses algorithms from list; sends back: choice + certificate + server nonce
3. client verifies certificate, extracts server's public key, generates pre_master_secret, encrypts with server's public key, sends to server
4. client and server independently compute encryption and MAC keys from pre_master_secret and nonces
5. client sends a MAC of all the handshake messages
6. server sends a MAC of all the handshake messages

Real SSL: handshaking (3)

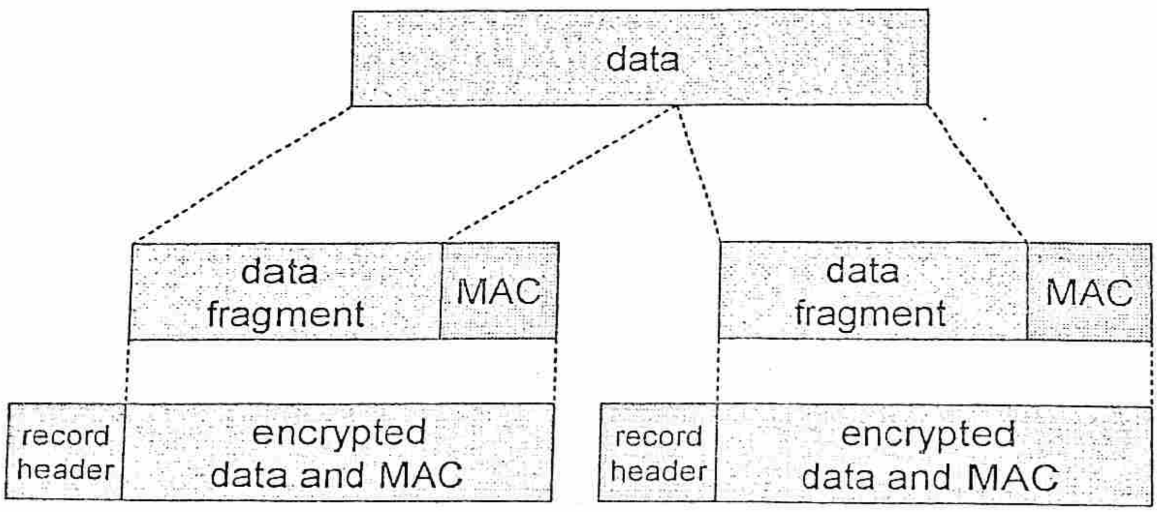
last 2 steps protect handshake from tampering

- client typically offers range of algorithms, some strong, some weak
- man-in-the middle could delete stronger algorithms from list
- last 2 steps prevent this
  - last two messages are encrypted

Real SSL: handshaking (4)

- why two random nonces?
- suppose Trudy sniffs all messages between Alice & Bob
- next day, Trudy sets up TCP connection with Bob, sends exact same sequence of records
  - Bob (Amazon) thinks Alice made two separate orders for the same thing
  - solution: Bob sends different random nonce for each connection. This causes encryption keys to be different on the two days
  - Trudy's messages will fail Bob's integrity check

SSL record protocol

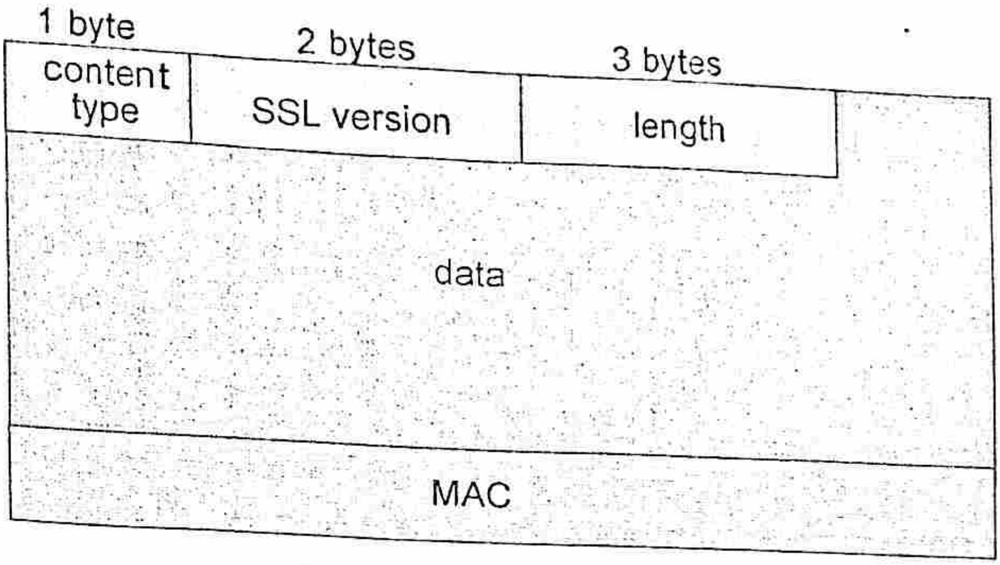


record header: content type; version; length

MAC: includes sequence number, MAC key M_x

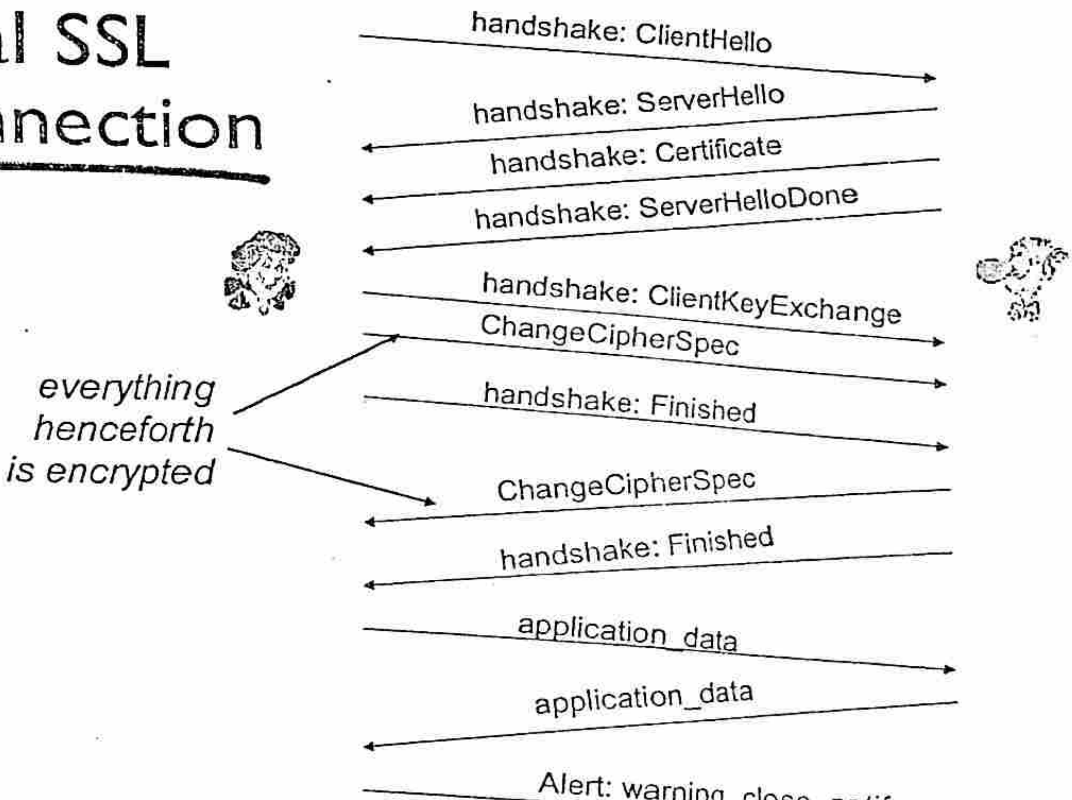
fragment: each SSL fragment 2^{14} bytes (~16 Kbytes)

SSL record format



data and MAC encrypted (symmetric algorithm)

Real SSL connection



TCP FIN follows

Key derivation

- client nonce, server nonce, and pre-master secret input into pseudo random-number generator.
  - produces master secret
- master secret and new nonces input into another random-number generator: "key block"
- key block sliced and diced:
  - client MAC key
  - server MAC key
  - client encryption key
  - server encryption key
  - client initialization vector (IV)
  - server initialization vector (IV)

Chapter 4

Computer Security

(Attacks)

Malicious Logic

Set of instructions that cause site`s security policy to be violated

Types of Malicious Logic

Trojan Horse

Program with an overt purpose (known to user) and a covert purpose (unknown to user) – Often called a Trojan Horse

Example NetBus

- Designed for Windows NT system (control remotely)
- Victim uploads and installs this
  - Usually disguised as a game program
- Acts as a server, accepting and executing commands for remote administrator
  - This includes intercepting keystrokes and mouse motions and sending them to attacker
  - Also allows attacker to upload, download files.

Replicating Trojan Horse

- Trojan horse that makes copies of itself
  - Also called propagating Trojan horse
- Hard to detect

Computer Viruses

- A computer virus is a program that inserts itself into one or more files and then performs some (possibly null) action.
- Two phases
  - insertion phase: The virus inserts itself into a file (or files)
  - Execution phase: The virus executes
- Insertion phase must be present
  - Does not need to be executed
  - Lehigh virus inserted itself into boot file only if boot file not infected

History

- Programmers for Apple II wrote some
  - Not called viruses; very experimental
- Fred Cohen
  - Graduate student who described them
  - Teacher (Adleman) named it "computer virus"
  - Tested idea on UNIX systems and UNIVAC 1108 system

Types of Viruses

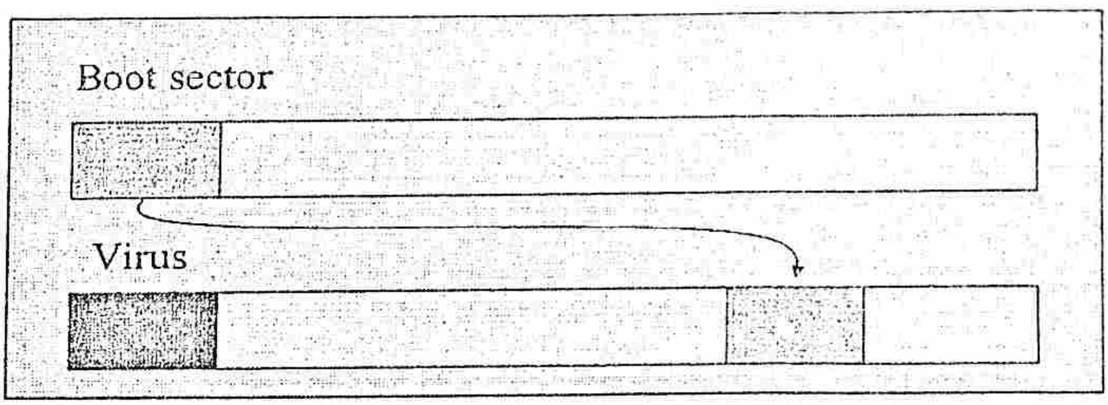
- Boot sector infectors
- Executable infectors
- Multipartite viruses
- TSR viruses
- Stealth viruses
- Encrypted viruses
- Polymorphic viruses
- Macro viruses

Boot sector infectors

- The boot sector is the part of a disk used to bootstrap the system.
- Code in a boot sector is executed when the system "sees" the disk for the first time.
- A virus that inserts itself into the boot sector of a disk – Section of disk containing code – Executed when system first "sees" the disk at boot time ...

Boot sector infectors

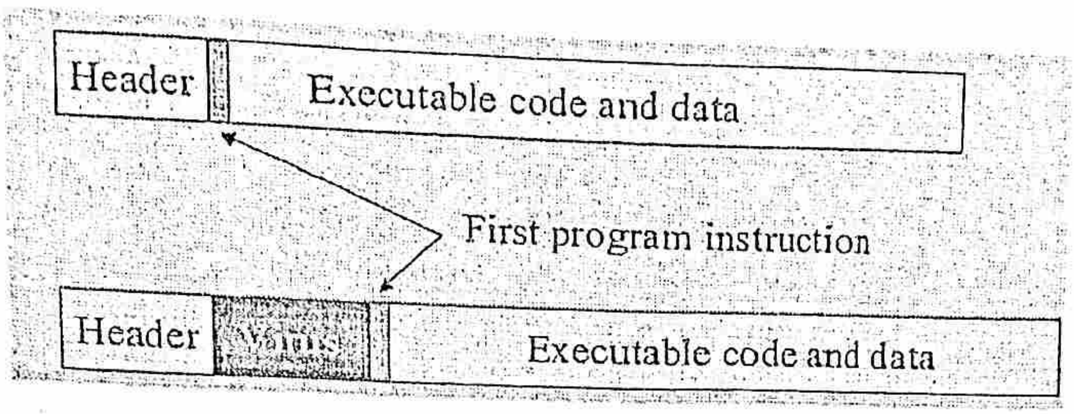
Infecting disks



1. Copy the old boot sector to alternative place;
2. Insert itself into the boot sector.

Executable infectors

- Triggered if an infected program is executed.



Executable infectors

- A virus that can infect either boot sectors or executable

- Typically, two parts :
  - One part boot sector infector
  - Other part executable infector

Multipartite Viruses

- A virus that infects executable programs
- Can infect either .EXE or .COM on PCs
- May put itself anywhere, fixing up binary so it is executed at some point.

TSR Viruses

- A virus that stays active in memory after the application (or bootstrapping, or disk mounting) is completed
- TSR is "Terminate and Stay Resident"

Computer Viruses (Cont'd)

- Polymorphic viruses
  - Change its form each time it inserts itself into another program.
- Stealth viruses
  - Conceal the infection of files
  - Make itself difficult to detect

Computer Viruses (Cont'd)

- Encrypted viruses
  - Encrypt itself with a random key
  - Avoid detection by anti-virus programs, which search for patterns of viruses.
- Macro viruses
  - Viruses composed of instructions that are interpreted, rather than executed.
- Examples: Word viruses, Email viruses
- MS Office suite is the most popular target.

Computer Worms

- A computer worm is a program that copies itself from one computer to another.
- Different from viruses
  - Viruses depend on other programs
  - Worms are usually standalone applications
  - Viruses usually trick people into propagating them
  - Worms can hack into vulnerable systems and spread without depending on others.

The Sapphire/Slammer Worm

- Facts about Sapphire/Slammer
  - Happened slightly before 5:30 UTC on Saturday, January 25, 2003.
  - The fastest worm in history.
  - Doubled in size every 8.5 seconds at the beginning
  - Infected more than 90% of vulnerable hosts within 10 minutes

The Sapphire/Slammer Worm (Cont'd)

- How does it find vulnerable computers?
  - Random scanning
  - Select IP addresses at random to infect
- How does it get into vulnerable computers?
  - Exploit a buffer overflow vulnerability in MS SQL Server or MSDE 2000
  - Vulnerability discovered in July 2002
- Why was it so fast?
  - Small: 376 bytes; a 404 byte UDP packet
  - Based on UDP

The Sapphire/Slammer Worm (Cont'd)

- What's its real impact (so far)?
  - Sapphire does not have a malicious payload
  - The Internet was saturated.
  - Too many hosts are infected and are trying to infect randomly selected hosts.

Rabbits, Bacteria

- A program that absorbs all or some class of resources
- Example: for UNIX system, shell commands:
  while true do

  mkdir x

  chdir x done
- Exhausts either disk space or file allocation table (inode) space

Logic Bombs

- A program that performs an action that violates the site security policy when some external event occurs
- Example: program that deletes company's payroll records when one particular record is deleted
- The "particular record" is usually that of the person writing the logic bomb
- Idea is if (when) he or she is fired, and the payroll record deleted, the company loses all those records

Defenses against Malicious Logic

- Type enforcement by human users
  - A program being written is considered data
  - A program must be changed into executable by a certifying authority before it's executed.
- Limiting the users' access domain
  - Idea: limit the objects that can be accessed by a malicious logic that assumes the user's privilege.
- Methods
  - Control information flow distances
    - Ex. Information cannot flow more than n times
  - Reduce the rights
  - Sand-boxing
    - Implicitly restrict process rights
    - Ex. Insert special instructions that cause traps whenever an instruction violates the security policy.

Defenses against Malicious Logic

- Inhibit users from sharing programs in different domains
  - An extreme: isolated domains

- Detect modified files
  - Using cryptographic check-sums to detect alteration of files

Chapter 5

(*Classes of Intruders – Cyber Criminals*)

Introduction



- Individuals or members of an organized crime group with a goal of financial reward
- Their activities may include:
  - Identity theft
  - Theft of financial credentials
  - Data theft

- Typically they are young, often Eastern European, Russian, or southeast Asian hackers, who do business on the Web
- They meet in underground forums to trade tips and data and coordinate attacks

Classes of Intruders – Activists

- Are either individuals, usually working as insiders, or members of a larger group of outsider attackers, who are motivated by social or political causes
- Also know as hacktivists
  - Skill level is often quite low
- Aim of their attacks is often to promote and publicize their cause typically through:
  - Website defacement
  - Denial of service attacks
  - Theft and distribution of data that results in negative publicity or compromise of their targets

Classes of Intruders – State-Sponsored

- Groups of hackers sponsored by governments to conduct espionage or sabotage activities.
- Also known as Advanced Persistent Threats (APTs) due to the covert nature and persistence over extended periods involved with any attacks in this class.
- Widespread nature and scope of these activities by a wide range of countries from China to the USA, UK, and their intelligence allies.

Classes of Intruders – Others

- Hackers with motivations other than those previously listed
- Include classic hackers or crackers who are motivated by technical challenge or by peer-group esteem and reputation
- Many of those responsible for discovering new categories of buffer overflow vulnerabilities could be regarded as members of this class.
- Given the wide availability of attack toolkits, there is a pool of "hobby hackers" using them to explore system and network security

Intruder Skill Levels – Apprentice

- Hackers with minimal technical skill who primarily use existing attack toolkits
- They likely comprise the largest number of attackers – including many criminal and activist attackers
- Given their use of existing known tools, these attackers are the easiest to defend against
- Also known as "script-kiddies" due to their use of existing scripts (tools)

Intruder Skill Levels – Journeyman

- Hackers with sufficient technical skills to modify and extend attack toolkits to use newly discovered, or purchased, vulnerabilities.
- They may be able to locate new vulnerabilities to exploit that are similar to some already known
- Hackers with such skills are likely found in all intruder classes
- Adapt tools for use by others

Intruder Skill Levels – Master

- Hackers with high-level technical skills capable of discovering brand new categories of vulnerabilities.
- Write new powerful attack toolkits
- Some of the better known classical hackers are of this level
- Some are employed by state-sponsored organizations
- Defending against these attacks is of the highest difficulty

Examples of Intrusion

- web server defacement
- guessing / cracking passwords
- copying databases containing credit card numbers
- viewing sensitive data without authorization
- running a packet sniffer
- distributing pirated software
- using an unsecured modem to access internal network
- impersonating an executive to get information
- using an unattended workstation

Intruder Behavior

- Target acquisition and information gathering
- Initial access
- Privilege escalation
- Information gathering or system exploit
- Maintaining access
- Covering tracks

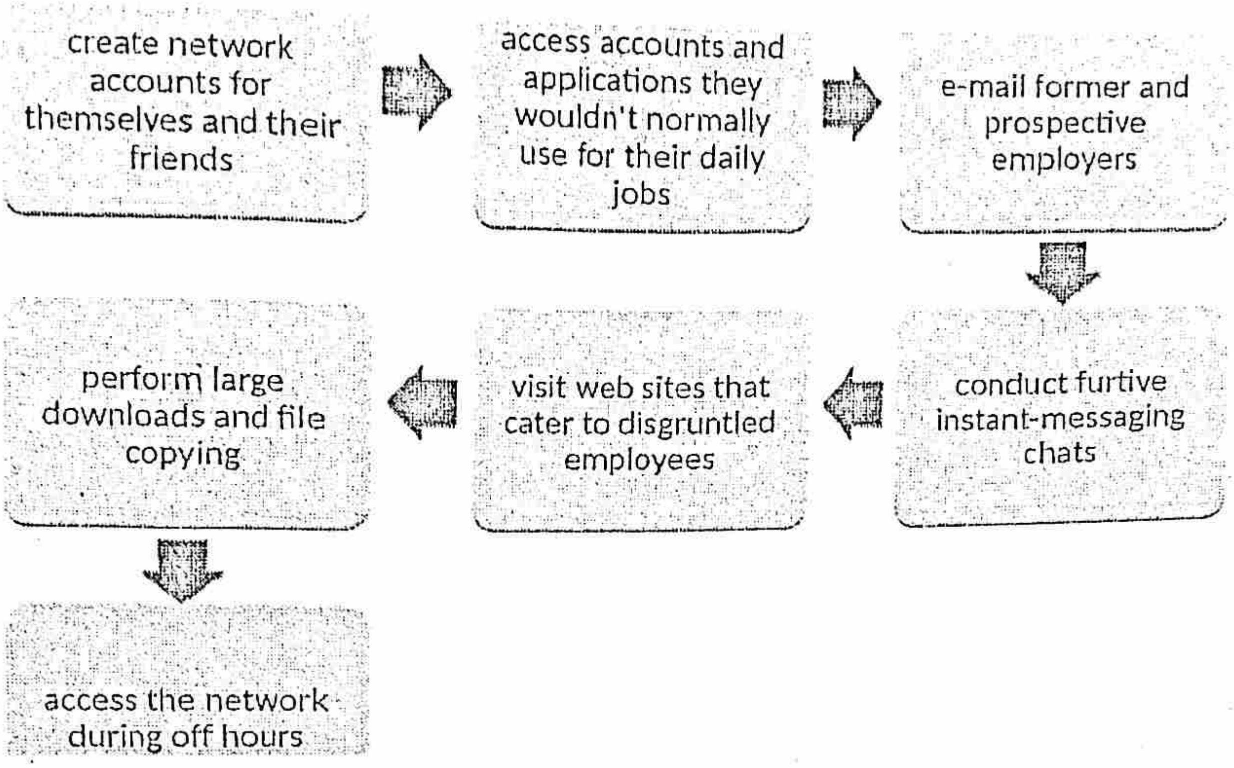
Hacker Patterns of Behavior

- select the target using IP lookup tools such as NSLookup, Dig, and others
- map network for accessible services using tools such as NMAP
- identify potentially vulnerable services (in this case, pcAnywhere)
- brute force (guess) pcAnywhere password
- install remote administration tool called DameWare
- wait for administrator to log on and capture his password
- use that password to access remainder of network

Criminal Enterprise Patterns of Behavior

- act quickly and precisely to make their activities harder to detect
- exploit perimeter via vulnerable ports
- use Trojan horses (hidden software) to leave back doors for re-entry
- use sniffers to capture passwords
- do not stick around until noticed

Internal Threat Patterns of Behavior



RFC 2828: Internet Security Glossary

- Security Intrusion: A security event, or a combination of multiple security events, that constitutes a security incident in which an intruder gains, or attempts to gain, access to a system (or system resource) without having authorization to do so.
- Intrusion Detection : A security service that monitors and analyzes system events for the purpose of finding, and providing real-time or near real-time warning of, attempts to access system resources in an unauthorized manner.

Intrusion Detection Systems (IDSs)

- host-based IDS
  -monitors the characteristics of a single host for suspicious activity

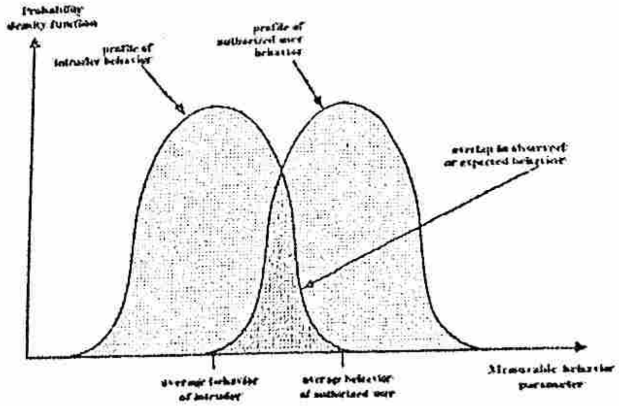
- network-based IDS
  -monitors network traffic and analyzes network, transport, and application protocols to identify suspicious activity

- Distributed or hybrid IDS
  -Combines information from a number of sensors, in a central analyzer that is able to better identify and respond to intrusion activity

IDS Principles

- assume intruder behavior differs from legitimate users

- overlap in behaviors causes problems
  -false positives
  -false negatives



IDS Requirements

- run continually
- be fault tolerant
- impose a minimal overhead on system
- configured according to system security policies
- adapt to changes in systems and users
- scale to monitor large numbers of systems
- Provide graceful degradation of service
- allow dynamic reconfiguration

Anomaly detection

- Involves the collection of data relating to the behavior of legitimate users over a period of time
- Current observed behavioris analyzed to determine whether this behavior is that of a legitimate user or that of an intruder

Signature/Heuristic detection

- Uses a set of known malicious data patterns or attack rules that are compared with current behavior
- Also known as misuse detection
- Can only identify known attacks for which it has patterns or rules

Anomaly Detection

- A variety of classification approaches are used:

Statistical

- Analysis of the observed behavior using univariate, multivariate, or time-series models of observed metrics

Knowledge based

- observed use an expert system that classifies observed behavior according to a set of rules that model legitimate behavior

Machine-learning

- Approaches automatically determine a suitable classification model from the training data using data mining techniques

Signature or Heuristic Detection

Signature approaches

- Match a large collection of known patterns of malicious data against data stored on a system or in transit over a network
- The signatures need to be large enough to minimize the false alarm rate, while still detecting a sufficiently large fraction of malicious data
- Widely used in anti-virus products, network traffic scanning proxies, and in NIDS

Rule-based heuristic identification

- Involves the use of rules for identifying known penetrations or penetrations that would exploit known weaknesses
- Rules can also be defined that identify suspicious behavior, even when the behavior is within the bounds of established patterns of usage
- Typically rules used are specific
- SNORT is an example of a rule-based NIDS



Host-Based IDS

- adds a specialized layer of security software to vulnerable or sensitive systems
- Can use either anomaly or signature and heuristic approaches
- monitors activity to detect suspicious behavior
  -primary purpose is to detect intrusions, log suspicious events, and send alerts
  -can detect both external and internal intrusions

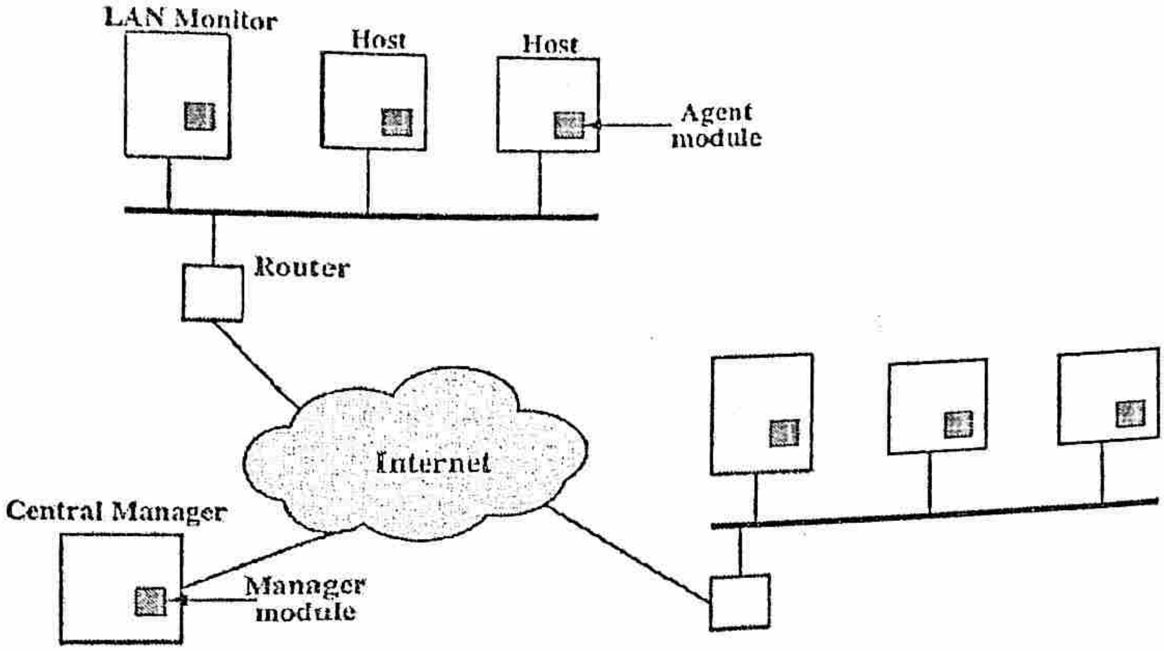
Data Sources and Sensors

A fundamental component of intrusion detection is the sensor that collects data

Common data sources include:

- System call traces
- Audit (log file) records
- File integrity checksums
- Registry access

Distributed Host-Based IDS



Network-Based IDS (NIDS)

- monitors traffic at selected points on a network
- examines traffic packet by packet in real or close to real time
- may examine network, transport, and/or application-level protocol activity
- comprised of a number of sensors, one or more servers for NIDS management functions, and one or more management consoles for the human interface
- analysis of traffic patterns may be done at the sensor, the management server or a combination of the two